第一章：

CPU在加电之后的第一条指令在哪：内存中，内存分为RAM和ROM（只读存储），

加电之后蹦到BIOS执行

启动，中断、异常和系统调用

操作系统一开始是放在硬盘上面的（DISK）

BIOS：基本I/O处理系统

Bootloader：加载OS，把硬盘放在内存中去

BIOS从特定地址开始执行CS:段寄存器 IP：指令寄存器 即PC=CS:IP

BIOS：支持基本输入输出，系统配置和开机自检、和系统的启动程序

POST加电自检，检查自身的设备能否正常工作

Bootloader一般放在硬盘的第一个引导扇区0X7c00（512字节）即加载程序

BIOS引导Blootloader，bootloader引导OS

Bootloader：把操作系统的代码和数据从硬盘加载到内存，跳转到操作系统的起始地址

系统调用：应用程序主动向操作系统发出服务请求（特殊的指令完成相应的功能）【异步或同步】

异常：非法指令或者其他坏的处理状态【同步】

中断：来源于外设【异步】

只有内核可以执行特权指令

中断：

中断表：不同的外设产生的中断有一个对应的中断号（中断事件的ID）

硬件，外设设置中断标记，CPU会产生一个中断号，操作系统根据中断号找到对应的处理历程

软件保存被打断的执行现场。便于后续的回复。根据cpu中断号查到中断处理例程，然后清楚中断标记，恢复中断前的状态。

异常：

异常ID号，保存产生异常的现场，比如产生异常的地址，寄存器的值。

根据异常来判断执行的操作：杀死产生异常的程序，或者恢复现场重新执行产生异常的指令

系统调用：

应用程序需要一个接口，系统调用接口

应用程序访问高层次API接口而不是直接系统调用

应用程序访问系统调用接口，触发从用户态到内核态的转变，内核态执行特权指令和IO指令，使得控制权从应用程序转移到操作系统

和函数调用的区别：

函数调用时是在一个栈空间完成了参数的传递和返回

系统调用时切换到内核时，也需要切换堆栈，用户态和内核态有各自的堆栈，更大的开销但是更安全

三者跨越操作系统的边界的开销：

执行时间上的开销

建立中断、异常、系统调用号和对应服务例程的映射关系的开销

建立内核堆栈

验证参数

内核态映射到用户态的地址空间